REMARKS

In the Office Action, the Examiner rejected Claims 1-30, which were all of the pending claims, over the prior art. Specifically, these claims were rejected either under 35 U.S.C. §102 as being fully anticipated by U.S. Patent 6,484,313 (Trowbridge, et al.), or under 35 U.S.C. §103 as being obvious over Trowbridge, et al. in view of U.S. Patent 5,978,585 (Crelier). Claims 2, 20, 21, 24 and 30 were further rejected under 35 U.S.C. §112 as being indefinite. In addition, the Examiner objected to the drawings, to the specification and to the Abstract.

In order to expedite this case, Claims 1-30 are being canceled and new Claims 31-34 are being added. Applicants reserve the right to file a continuation application to continue the prosecution of Claims 1-30.

Each of the new claims 31-34 is an independent claim. For the reasons set forth below, new Claims 31-34 patentably distinguish over the prior art and are allowable.

In objecting to the drawings, the Examiner noted that "412" is missing from the drawings, and "400" is shown in the drawings but is not referenced in the specification. In response, Applicants are herein amending the specification, at page 9, line 23, to add the reference number "400," and at page 10, line 16, to delete the reference number "412." With these changes, it is believed that every reference number in the drawings is also referred to in the specification, and every reference number mentioned in the description is also shown in the drawings. The Examiner is, accordingly, respectfully requested to reconsider and to withdraw the objection to the drawings.

In response to the Examiner's objection to the specification, Applicants are herein amending the specification to delete the hyperlink on page 6, to capitalize the word "JAVA,"

and to accompany that term with generic terminology. Also, the brackets are being deleted from page 17, line 21.

In the Office Action, the Examiner indicated, in paragraph 6, that the Abstract was objected to because of the use of brackets. It is believed that this, in fact is a reference to the brackets on page 17, and it is noted that there are no brackets in the Abstract.

In view of the above, the Examiner is also asked to reconsider and to withdraw the objections to the specification and to the Abstract.

With respect to the references cited by the Examiner, Applicants note that there are several features of the method disclosed in the present application that are not disclosed or suggested by the cited references. Generally, these features relate to security, linkage at run time, invalidating the code, and maintaining a preferred compliance.

With specific regard to Claim 31, in the method of Trowbridge, et al., beginning at Column 9, line 60 and on, mention is made of "versioning information" about the environment the code is compiled in being maintained, including classes loaded at the time of compilation. Mention is also made of classes being recompiled when this versioning information disagrees with what is present in the current execution environment where statically compiled code is being loaded for execution. The method of Claim 31 differs from that described by Trowbridge, et al. in that the method of Claim 31 takes actions specifically directed towards maintaining the security of the generated code, over and above the verification described by Trowbridge, et al. These steps, which are directed towards maintaining the security of the code, allow code generated by the processes to be exchanged between machines.

Claim 31 differs from Crelier, et al. in the method of Claim 1 relies on more than time-stamps to determine the validity of a file, in particular we take actions to maintain the ability to validity checks even when files are exchanged through a hostile environment, where, for example, time-stamps might be maliciously altered.

With respect to Claim 32, in the method of Trowbridge, et al., beginning in Column 11, line 32 and continuing, Column 12, line 33 and continuing, Column 13, line 12 and continuing, a process of creating "cookies" that contain requests for information not available at compile time are described. The information requested includes the concrete representation of data structures at runtime, addresses not available at compile time, and other unspecified information. When this information is obtained, the cookies are updated (column 13, line 9) and overwritten with this information. Since the cookie is described in Trowbridge as a table separate from the generated code, this implies an indirection during the execution of the code. For example, to access a variable whose address was not known at compile time will require the executing program to access the cookie table, access the address contained in the cookie, and then access the actual datum. In the present invention, the request for information contained in the qsi contains a mapping into the actual generated code. This code may be updated during loaded (although the invention is not limited to performing the update only at load time) and at run-time indirections are eliminated. Thus, to repeat the example above, the executing code needs to only perform a "load immediate" instruction on the address of the datum, and then access the datum – a superior process to that described in Trowbridge, et al.

New Claim 33 is directed to a method for describing dependencies (described as "fine grained dependencies" in the preferred embodiment) that allows us to securely and correctly accept as valid for reused code that depends on a class that has changed in its implementation

of functions, but has not changed the interfaces of functions that the dependent code relies on. This allows us to fully exploit the concept of "binary compatibility" to maximize the amount of reuse of statically compiled code. The method described in Trowbridge, et al. works only on version numbers, time stamps, OS versions, and so forth, which are inherently less fine-grained than method of Claim 33, and therefore will require more recompilations than the method of Claim 33.

With respect to new Claim 34, in Trowbridge, et al., Column 9, line 65, the alternatives available to the execution engine when the precompiled code fails to pass the various tests of the methods are to (1) not execute the code; or (2) to compile the code into native code (possibly in a background process) and then compile it into native code for execution. The invention defined by Claim 34, describes how an execution engine can use an interpreter to execute the code. This allows the execution engine to not contain a compiler, which in turn allows significantly less storage to be used by the execution engine, which is a significant benefit in cost sensitive, power sensitive and/or space sensitive environments such as small handheld devices and embedded devices. Because files will be invalidated rarely, the cost of interpreting, rather than executing native code will, on average, be tiny compared to the benefits of not including a compiler in the execution engine.

The other references have been reviewed and they are not believed to be any more pertinent to the patentability of new Claims 31-34 than Trowbridge, et al.

Because of the above-discussed differences between Claims 31-34 and the prior art, and because of the advantages associated with those differences, Claims 31-34 patentably distinguish over the prior art and are allowable.

For the reasons advanced above, the Examiner is requested to reconsider and to withdraw the objections to the specification, the drawings and the Abstract, and to allow Claims 31-34. If the Examiner believes that a telephone conference with Applicants' Attorneys would be advantageous to the disposition of this case, the Examiner is asked to telephone the undersigned.

Respectfully submitted,

John S Sensny

John S. Sensny
Registration No. 28,757

Scully, Scott, Murphy & Presser
400 Garden City Plaza
Garden City, New York 11530
(516) 742-4343
JSS:gc